Control of input/output discrete-event systems

M. Petreczky, R. Theunissen, R. Su, D.A. van Beek, J. H. van Schuppen, J.E. Rooda
Eindhoven University of Technology P.O. Box 513, 5600 MB Eindhoven, The Netherlands
Centrum voor Wiskunde en Informatica (CWI) P.O.Box 94079, 1090GB Amsterdam, The Netherlands
{M.Petreczky, R.J.M.Theunissen, R.Su, D.A.v.Beek, J.E.Rooda}@tue.nl, J.H.van.Schuppen@cwi.nl

***Abstract*— A class of control problems for discrete-event systems is proposed, inspired by applications in the domain of high-tech systems. The control problem asks for controllers which generate control inputs based on the outputs. We formalize the above control problem, whereby the plant behavior is modeled as an input-output relation recognizable by a rational transducer, and the controller is modeled as a sequential map realizable by a Moore-automaton. The control objective is formalized as a language over the alphabet of internal (unobservable) events generated by the plant. We propose a solution to the control problem above by reducing it to a Ramadge-Wonham control problem with partial observations.** [1]

I. Introduction

Motivated by applications in the area of high-tech systems, in particular, printers [24] and MRI scanners [25], we consider the following control problem.

Control problem The plant we are interested in changes its behavior under control inputs and external inputs and generates outputs and internal events. The control inputs and external inputs, the outputs and the internal events are all sequences of symbols from a finite alphabet. The external inputs are imposed by the environment (user) and the control inputs are the ones which can be used by the controller to influence the plant behavior. In addition, the external inputs, control inputs, outputs and internal events are allowed to happen simultaneously. In other words, the systems of interest react to any pair of control inputs and external inputs by generating outputs and internal events. A *controller* reads the outputs generated by the plant and the external inputs generated by the environment/user and *generates* a control input for each possible history. It is activated on sampling times or whenever an event occurs. The objective is to ensure that the closed-loop system generates sequences which belong to the language of *control requirements*.

Contribution of the paper The contribution of the paper is twofold.

(1) formalization of the control problem We formalize the control problem sketched above. We model the distinction between inputs and outputs explicitly, i.e. the plant is a dynamical system which under the influence of inputs generates outputs. Similarly, we model controllers as dynamical systems which read sequences of outputs and generate control symbols which are fed back to the plant. Mathematically, the *external behavior* of the plant is an *input-output relation* mapping finite strings to finite strings. The underlying *state-space representation* of the plant is a *transducer* [9], [2]. The mathematical model of the controller is a *sequential input-output function*, and the underlying *state-space* representation is a *finite-state deterministic Moore-automaton* [9], [11].

(2) rigorous solution of the control problem We show that despite apparent differences, the formulated control problem can be transformed to a classical Ramadge-Wonham (abbreviated as **RW**) control problem and solved using RW theory. The latter contribution illustrates vividly the versatility of the classical RW framework. The proposed transformation is theoretically sound and computable, but it works only under additional assumptions. The general case can be treated using Rabin- or parity-games [14]. We defer the treatment of the general case to another paper.

Informally, the proposed solution of the control problem consist of the following steps:

1) Provide a model of the plant as a transducer.
2) Provide a model the control requirements and finite-state automaton.
3) Construct a finite-state automaton from the transducer of the plant. This finite-state automaton will play the role of the new plant model of the corresponding RW control problem.
4) Likewise, transform the finite-state automaton of the requirements to a finite-state automaton (defined over a different alphabet) which will be interpreted as the automaton recognizing the requirements of the corresponding RW control problem.
5) Solve the corresponding RW control problem (with partial observations) for the plant automaton and the automaton of the control requirements obtained in the previous steps and obtain a (not necessarily maximally permissive) supervisor.
6) Extract from the finite-state automaton implementing the supervisor the Moore-automaton of the controller.

Motivation for the new formalism The motivation for using a separate formalism instead of the RW framework is the following.

(1) The framework explicitly formalizes the distinction between inputs and outputs, and which events are generated by the plant, which by the environment and which by the controller. The RW framework leaves these issues unspecified. This makes the RW framework very flexible, but may

[1]This work was supported by the ITEA project Twins 05004; the DARWIN project at Philips Healthcare under the responsibility of the Embedded Systems Institute, partially supported by the Netherlands Ministry of Economic Affairs under the BSIK program.

lead to difficulties in applications, [1], [8], [5], [10], [7], [20].

(2) Notice that not every instance of the proposed control problem can be solved using RW theory. In fact, game theory [14] can also be used to solve the control problem, even for cases when RW theory cannot be applied. Hence, it makes sense to formalize the proposed control problem separately.

Related work To the best of our knowledge, the presented results are new. A more complete version of this paper can be found in [23]. Many of the challenges regarding application of RW theory, including the need for explicit modeling of inputs and outputs, were already mentioned in [1], [8], [5], [10], [7], [20]. With respect to [1], we allow unobservable internal events and our way of modeling inputs and outputs explicitly is closer to classical control and it is perhaps more intuitive for the problem at hand. However, [1] also addresses communication delays, which topic is absent from this paper. With respect to [19], [13], [4], [16], the main difference is that we allow unobservable internal events and uncontrollable inputs and our control problem is not based on enabling/disabling events. The control problem of this paper is completely different from [8]. Extensions of RW theory where the supervisor forces controllable events was investigated in [10], [7], [3], [12], [15]. However, in these papers partial observations and explicit input-output modeling were not considered, and the framework seems to be further from the physical reality of high-tech systems than the one of this paper. Contrary to [7], [15] the problem of non-blockingness is not relevant in our case, due to the specific problem formulation. In [22] input/output discrete-event systems were introduced in order to facilitate hierarchical control design. In contrast, in this paper we are not interested in hierarchical control. This leads to several subtle differences between our framework and that of [22]. In addition, we use a completely different mathematical language to formalize the control problem. The problem of extracting deterministic supervisors was addressed in [21], however there the explicit modeling of inputs and outputs and partial observations were not addressed. Automata with inputs and outputs have appeared in the context of model matching problem [6]. Model matching is related to, but different from the control problem of this paper. Other aspects of input-output modeling and automata were discussed in [17]. Automata with inputs and outputs is a classical topic, see [9], [11], and [18].

Outline of the paper In §II we formally state the discrete-event control problem we are interested in. In §III we state what kind of finite representation of the plant behavior and of the control requirement specification is necessary for solving the control problem. In §IV we describe how to solve our control problem using RW theory.

II. Problem formulation

The goal of this section is to formulate the control problem studied in this paper. In §II-A the notation used in the paper is reviewed. In §II-B the formal statement of the control problem of interest is presented.

A. Notation

We use the standard notation and terminology from automata theory [9], [11]. Let Σ be a finite set, referred to as the *alphabet*. Σ^* denotes the set of finite *strings (words)* of elements of Σ, i.e. an element of Σ^* is a sequence $w = a_1 a_2 \cdots a_k$, where $a_1, a_2, \ldots, a_k \in \Sigma$, and $k \geq 0$; k is *the length of* w and it is denoted by $|w|$. If $k = 0$, then w is the empty word, denoted by ϵ. The concatenation of two words v and w is denoted by vw. An *infinite (ω-) word* over Σ is an infinite sequence $w = a_1 a_2 \cdots a_k \cdots$ with $a_i \in \Sigma$, $i \in \mathbb{N}$. The set of infinite words is denoted by Σ^ω.

A *language* over Σ is a set of finite strings (words) over Σ. For any (in)finite word w, and for any $i \in \mathbb{N}$ (in case w is finite word, for any $i \in \mathbb{N}$ such that $i \leq |w|$), $w_{1:i}$ denotes the finite word formed by the first i letters of w, i.e. $w_{1:i} = a_1 a_2 \cdots a_i$. If $i = 0$, then $w_{1:i}$ is the empty word ϵ.

For any word $w \in \Sigma^* \cup \Sigma^\omega$, a finite word $p \in \Sigma^*$ is a *prefix* of w, if there exists an index $i \in \mathbb{N}$, such that $w_{1:i} = p$. If $K \subseteq \Sigma^*$, then $\lim(K) \subseteq \Sigma^\omega$ is the set of all infinite words, infinitely many prefixes of which belong to K, i.e.

$$\lim(K) = \{w \in \Sigma^\omega \mid \exists \{k_i \in \mathbb{N}\}_{i \in \mathbb{N}} : \\ \forall i \in \mathbb{N} : (k_{i+1} > k_i \text{ and } w_{1:k_i} \in K)\}$$

If $L \subseteq \Sigma^* \cup \Sigma^\omega$, then the *prefix closure* of L is denoted by $\bar{L}$ and is defined by $\bar{L} = \{p \in \Sigma^* \mid \exists v \in L : p \text{ is a prefix of } v\}$; L is called *prefix closed*, if $\bar{L} = L$.

A map $\theta : X^* \to Y^*$, where X and Y are finite alphabets, is called a *morphism*, if θ preserves the empty sequence and concatenation, i.e. $\theta(\epsilon) = \epsilon$ and $\theta(wv) = \theta(w)\theta(v)$.

Recall from [9], [11] that a *Moore-automaton* is a tuple $A = (Q, I, Y, \delta, \lambda, q_0)$ where Q is the finite *state-space* of A, I is the *input alphabet* of A, Y is the *output alphabet* of A, $\delta : Q \times I \to Q$ is the *state-transition map* of A, $\lambda : Q \to Y$ is the *readout map* of A, and $q_0 \in Q$ is the *initial state* of A. The Moore-automaton A is a *realization* of a map $\phi : I^* \to Y$, if for all $w = u_1 u_2 \cdots u_k \in I^*$, $k \geq 0$ and $u_1, u_2, \ldots, u_k \in I$, $\phi(w) = \lambda(q_k)$ where $q_i = \delta(q_{i-1}, u_i)$ for all $i = 1, 2, \ldots, k$. The map ϕ is *realizable* by a Moore-automaton, if there exists a Moore-automaton which is a realization of ϕ.

B. Input-output control problem

The systems (plants) of interest have four types of signals; *control inputs* from U, *external inputs* from D, *observable outputs* O, and *internal events* from E_i. Only the elements of U and D are capable of changing the dynamics of the system. The appearance of symbols from O and E_i *may indicate* an occurrence of a state-transition, but is *does not trigger a state-transition itself*. Typical elements of U could be to switch an engine on/off, typical elements of D are events such as a button pressed, error message has arrived, etc. Typical elements of O are sensor data, typical elements of E_i are invisible events which are needed for the specification of the control objectives. We will use the *notion of sequential input-output relations* to formalize the the input-output behavior of the plant.

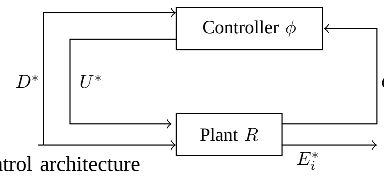

Fig. 1. Control architecture

Definition 1 (Sequential input-output relation): A multi-valued map $R : (U \times D)^* \to 2^{O^* \times E_i^*}$ is called a *sequential input-output relation*, if the following conditions are satisfied

1) $R(\epsilon) = \{(\epsilon, \epsilon)\}$, and for all $s \in (U \times D)^*$, $R(s)$ is a non-empty finite set.
2) For all $s \in (U \times D)^*$, and for all $(\underline{o}, \underline{\hat{o}}) \in R(s)$, with $\underline{o} \in O^*$ and $\underline{\hat{o}} \in E_i^*$, the length of s and $\underline{o}$ are the same, i.e. $|s| = |\underline{o}|$.
3) R is *prefix preserving*, i.e. for each $s \in (U \times D)^*$ and $(u, d) \in (U \times D)$, the relation $(\underline{o}, \underline{\hat{o}}) \in R(s(u, d))$ implies that there exist strings $\underline{o}_1 \in O^*$, $\underline{\hat{o}}_1, \underline{\hat{o}}_2 \in E_i^*$ and a letter $o_2 \in O$ such that $(\underline{o}_1, \underline{\hat{o}}_1) \in R(s)$ and $\underline{o} = \underline{o}_1 o_2$ and $\underline{\hat{o}} = \underline{\hat{o}}_1 \underline{\hat{o}}_2$.

Notice that the value $R(s)$ of R above is non-empty for all sequences s. Notice that if the length of the control input and external input sequence increases, so does the length of the sequence of the observable outputs produced by the plant modeled by R. In fact, the length of the observable output sequence is the same as the length of the input sequence. However, the length of the sequence of internal events need not increase. If R is a function and R preserves the length of the argument in its E_i^*-valued component as well, then R is simply a sequential function [9], [11], generated by a (possibly infinite-state) Mealy-automaton [9], [11].

The task of a would-be controller is to generate control inputs based on past outputs and external inputs, such that the control objectives are met.

Definition 2 (Sequential controllers): A *sequential controller* is a map of the form $\phi : (D \times O)^* \to U$ such that ϕ is realizable by a Moore-automaton with input alphabet $D \times O$ and output alphabet U.

That is, a sequential controller is simply a dynamical system, which reads the external inputs and the output of the plant, updates its internal state and generates a control input.

The structure of the controller explains the requirement that the observable output sequence generated by the plant should be of the same length as the input sequence. The elements of O and D represent the information available to the controller at activation times, and hence their number is tied to the number of times the controller was activated. In contrast, the events from E_i are never used for control, they appear only in the specification. Next, we define the behavior of the closed-loop system.

Definition 3 (Feedback): Let $R : (U \times D)^* \to 2^{O^* \times E_i^*}$ be the sequential input-output relation of the plant and $\phi : (D \times O)^* \to U$ be a sequential controller. The behavior of the feedback interconnection of R with ϕ is the map $B(R/\phi) : D^* \to 2^{E_i^*}$, defined as follows. Fix sequence of external inputs $s = d_1 d_2 \cdots d_k \in D^*$, $d_1, d_2, \ldots, d_k \in D$, $k \geq 0$. If $k = 0$, i.e. $s = \epsilon$ is the empty sequence, then let $B(R/\phi)(s) = \{\epsilon\}$. If $k > 0$, then let $B(R/\phi)(s)$ be the set of words of the form

$$\underline{\hat{o}} = \hat{o}_1 \hat{o}_2 \cdots \hat{o}_k \in E_i^* \tag{1}$$

where $\hat{o}_i \in E_i^*$, $i = 1, \ldots, k$, and there exist control inputs $u_i \in U$, and outputs $o_i \in O$, $i = 1, 2, \ldots, k$ such that

$$\begin{aligned} &(o_1 o_2 \cdots o_i, \hat{o}_1 \hat{o}_2 \cdots \hat{o}_i) \in R((u_1, d_1)(u_2, d_2) \cdots (u_i, d_i)) \\ &u_i = \phi((d_1, o_1)(d_2, o_2) \cdots (d_{i-1}, o_{i-1})) \end{aligned} \tag{2}$$

Here, for $i = 1$ $(d_1, o_1)(d_2, o_2) \cdots (d_{i-1}, o_{i-1})$ is idenfitied with the empty sequence ϵ and hence $u_1 = \phi(\epsilon)$.

Intuitively, the external inputs from D are taken as the inputs of the closed-loop system, and the internal events as outputs. Next, we define the notion of the unobservable language of the closed-loop system. The latter is used to define when the closed-loop system meets the control objectives. Recall from §II-A that $w_{1:i}$ denotes the prefix of a (possibly infinite) word w, formed by the first i letters.

Definition 4 (Language of the closed-loop system): Define the *closed-loop language* $L(R/\phi) \subseteq E_i^* \cup E_i^\omega$ of the interconnection of R and ϕ as follows.

1) $\underline{\hat{o}} \in E_i^*$ belongs to $L(R/\phi)$ if there exists an infinite word $v \in D^\omega$ such that starting from some index $N \in \mathbb{N}$, for each index $i \geq N$, $\underline{\hat{o}} \in B(R/\phi)(v_{1:i})$.
2) $\underline{\hat{o}} \in E_i^\omega$ belongs to $L(R/\phi)$ if there exists an infinite word $v \in D^\omega$ and an infinite sequence of indices $k_0 \leq k_1 \leq \cdots k_i \leq \cdots$ such that $\sup_{i \in \mathbb{N}} k_i = \infty$ and for each $i \in \mathbb{N}$, $\underline{\hat{o}}_{1:k_i} \in B(R/\phi)(v_{1:i})$.

The language $L(R/\phi)$ is the set of all (in)finite sequences of internal events generated by the closed-loop system. Next we formulate the control problem studied in this paper.

Problem 1 (Discrete-event control problem): For a specified *sequential input-output relation* R modeling the plant, and for a *specification language* $K \subseteq E_i^* \cup E_i^\omega$ modeling the control requirements, find a sequential controller ϕ such that $L(R/\phi) \subseteq K$.

That is, the internal events generated by the closed-loop system must belong to the specification language K containing both finite and infinite words.

Remark 1 (Restriction on U, D and O): Notice that the elements of U, D and O are not explicitly included in the specification language K. However, conditions on control inputs, external inputs and observable outputs can be incorporated into our framework as follows. Modify the plant model by adding the current control input, and/or external inputs, and/or observable output as new components of the currently generated internal event, and then adapt the specification language K accordingly.

III. Finitary representation

In this section the assumptions on the finite-state representation the plant behavior, and of the specification language of the control objectives will be stated. To this end, in §III-A we recall the necessary notions from automata theory. In §III-B we state the assumptions on the finite-state representations.

A. Review of automata theory: monoids and transducers

Recall from [9], [2] that a *monoid* M is a (not necessarily finite) semi-group with a unit element which is denoted by 1_M, or simply 1, if M is clear from the context. Recall from [2], [9] that a transducer is a non-deterministic finite-state automaton with inputs and outputs from monoids.

Definition 5 (Transducer, [2], [9]): A transducer is a tuple $T = (Q, M_i, M_o, E, F, q_0)$ where

- Q is a finite set of states
- M_i is the monoid of inputs
- M_o is the monoid of outputs.
- $E \subseteq Q \times M_i \times M_o \times Q$ is a finite set, describing the state-transition relation.
- $F \subseteq Q$ is the finite set of accepting states
- $q_0 \in Q$ is the initial state

Definition 6 (Accepting run, [2], [9]): A pair $(u, y) \in M_i \times M_o$ is *accepted* by T if there exist elements $u_i \in M_i$ and $y_i \in M_o$ and states $q_i \in Q$, $i = 1, 2, \ldots, k$ for some $k \geq 0$ such that $(q_i, (u_i, y_i), q_{i+1}) \in E$ for $i = 0, 1, \ldots, k-1$, $q_k \in F$ and $u = u_1 u_2 \cdots u_k$ and $y = y_1 y_2 \cdots y_k$.

Definition 7 (Relation accepted by a transducer, [2], [9]): The relation $R \subseteq M_i \times M_o$ is *accepted by* T, and it is denoted by $R(T)$, if R consists of precisely those pairs $(u, y) \in M_i \times M_o$ which are accepted by T.

Definition 8 (Rationality of a relation): A relation $R \subseteq M_i \times M_o$ is called *rational*, if there exists a transducer T such that R is the relation accepted by T, i.e. $R(T) = R$.

B. Finite representation for plant and specification

Below we state the assumptions on the finite representation of the plant and of the specification. As to the plant, we require that the sequential input-output map R of the plant has a representation by a transducer of a specific type. To this end, notice that $O^* \times E_i^*$ is a monoid with respect to component-wise concatenation as multiplication and with the unit element (ϵ, ϵ). Similarly $(U \times D)^*$ is a monoid with the concatenation as multiplication and the empty word ϵ as the unit element.

Definition 9: The sequential input-output relation $R : (U \times D)^* \to 2^{O^* \times E_i^*}$ is called *rational*, if it is rational when identified with the relation $\widetilde{R} \subseteq M_i \times M_o$, where $M_i = (U \times D)^*$ is viewed as the input monoid, and $M_o = (O^* \times E_i^*)$ viewed as the output monoid, and $(s, (\underline{o}, \hat{\underline{o}})) \in \widetilde{R}$ if and only if $(\underline{o}, \hat{\underline{o}}) \in R(s)$. We will say that a transducer accepts R, if it accepts $\widetilde{R}$ according to Definition 8.
If T is a transducer accepting R, then the labels of the state-transition graph of T are labeled by 3 tuples of words $(U \times D)^* \times O^* \times E_i^*$. In order to simplify the discussion, we will require that the input-output relation R of the plant can be accepted by a transducer, whose state-transition graph is labeled by $(U \times D) \times (O \times E_i^*)$ and whose set of accepting states is the whole state-space. Note that the transducer above can be thought of as a non-deterministic Mealy machine [9], [11]. As to the specification language K, we require that its component consisting of words of inifnite length is a limit of a regular language, and its component made up of words of finite length is a regular language. To sum up,

Assumption 1 (Finite-state assumption): **Plant**
The sequential input-output relation R is rational, moreover, there exists a transducer $T = (Q, (U \times D)^*, (O^* \times E_i^*), E, F, q_0)$ accepting R such that

- $F = Q$, i.e. all states are accepting,
- $E \subseteq Q \times (U \times D) \times (O \times E_i^*) \times Q$, i.e. the state-transitions are labeled by of *letters* from U, D and O and *sequences* from E_i^*,
- for each $q \in Q$ and $(u, d) \in U \times D$ there exist $q^{'} \in Q$, $o \in O$ and $\hat{\underline{o}} \in E_i^*$ such that $(q, (u, d), (o, \hat{\underline{o}}), q^{'}) \in E$.

Specification language
$K = K_{safe} \cup \lim(K_{safe})$ where $K_{safe} \subseteq E_i^*$ is regular and prefix-closed language.
The second assumption on K essentially says that the control requirement should be a safety specification, i.e. the system should always produce words belonging to K_{safe}.

IV. Solution through supervisory control

In this section we show that Problem 1 can be reduced to classical RW control problem as follows.

1) Transform an instance of Problem 1 to an instance of a RW control problem with partial observations.
2) Synthesize a supervisor for the RW control problem using the well-known tools and algorithms from [26].
3) Extract from the supervisor a sequential controller which solves the original problem instance.

Below we elaborate on each step above and show that they are computationally effective.

A. From Problem 1 to a RW problem

First, we define the plant language of the RW control problem corresponding to Problem 1.

Definition 10 (Plant-language): The *plant language* $L_R \subseteq (U \cup D \cup O \cup E_i)^*$ associated with the sequential input-output map R is defined as

$$L_R = \{(u_0 d_1 o_1 \hat{o}_1)(u_1 d_2 o_2 \hat{o}_2) \cdots (u_{k-1} d_k o_k \hat{o}_k) u_k \mid u_i \in U,$$
$$u_0 \in U, d_i \in D, o_i \in O, \hat{o}_i \in E_i^*,$$
$$(o_1 o_2 \cdots o_i, \hat{o}_1 \hat{o}_2 \cdots \hat{o}_i) \in R((u_0, d_1)(u_1, d_2) \cdots (u_{i-1}, d_i))$$
$$\text{for } i = 1, 2, \ldots, k, k \geq 0\}$$

That is, the plant language L_R consists of strings, which are made up of groups of symbols, first element of which is the control input, the second one is the external input, the third one is the output and the fourth one is the sequence of internal events produced by the plant. That is, if $w \in L_R$, then for some $k \geq 0$, w can be decomposed as $w = w_1 w_2 \cdots w_k u_k$ where $w_i = u_{i-1} d_i o_i \hat{o}_i$ such that d_i is the external input at step i, o_i is the observable output produced at step i, $\hat{o}_i$ is the sequence of internal events at step i and finally u_{i-1} is the input received at step i. Finally, u_k is the input received at step $k+1$. The intuition behind L_R is that its words keep recognizable the basic cycle of reading input and producing output. Note that any other ordering of events within w_i could have been taken, as long as this order if fixed for all words of the plant language. However, the chosen ordering makes the application of Ramadge-Wonham theory easier. The following proposition follows easily from standard automata theory [9], [2].

Proposition 1 (Regularity of L_R): If R satisfies Assumption 1, then L_R is regular, and a deterministic automaton recognizing L_R can be computed from any transducer T accepting R, if T satisfies the condition of Assumption 1.

Proof: [Proof of Proposition 1] Denote by $\Sigma = (U \cup D \cup O \cup E_i)$ the alphabet over which L_R is defined. Recall from Definition 9 the monoids $M_i = (U \times D)^*$ and $M_o = O^* \times E_i^*$. Let $T = (Q, M_i, M_o, E, F, q_0)$ be a transducer accepting R which satisfies the conditions of Assumption 1. Based on T we will construct a non-deterministic finite state automaton $\mathcal{A}$ accepting L_R. By Assumption 1, the state-transition relation E is of the form $E \subseteq Q \times (U \times D) \times (O \times E_i^*) \times Q$, and that $F = Q$, i.e. all the states are accepting. Recall that E is a finite set, and hence the set $\mathcal{U} = \{v \in E_i^* \mid \exists (u,d,o) \in (U \times D \times O), q_1, q_2 \in Q : (q_1, (u,d), (o,v), q_2) \in E\}$ is finite. Denote by $\bar{\mathcal{U}}$ the set of prefixes of elements of $\mathcal{U}$. Notice that $\bar{\mathcal{U}}$ is a finite set as well. Define the sets $Q_0 = Q \times U$, $Q_1 = Q_0 \times D$, $Q_2 = Q_1 \times O \times \bar{\mathcal{U}}$. The sets Q_i, $i = 0,1,2$ are disjoint and finite. Let $q_\perp$ be a symbol not in $\bigcup_{i=0}^{2} Q_i$. Define the set $\hat{Q} = \{q_\perp\} \cup \bigcup_{i=0}^{2} Q_i$. Define the relation $\delta \subseteq \hat{Q} \times \Sigma \times \hat{Q}$ as follows. $(\hat{q}_1, e, \hat{q}_2) \in \delta$ if and only if one of the following conditions hold.

1) $\hat{q}_1 = q_\perp$ and $e = u \in U$ and $\hat{q}_2 = (q_0, u) \in Q_0$
2) $\hat{q}_1 = (q,u) \in Q_0$ and $e = d \in D$ and $\hat{q}_2 = (q,u,d) \in Q_1$
3) $\hat{q}_1 = (q,u,d) \in Q_1$ and $e = o \in O$ and $\hat{q}_2 = (q,u,d,o,\epsilon) \in Q_2$
4) $\hat{q}_1 = (q,u,d,o,v) \in Q_2$ and $e \in E_i$ and $ve \in \bar{\mathcal{U}}$ and $\hat{q}_2 = (q,u,d,o,ve) \in Q_2$.
5) $\hat{q}_1 = (q,u,d,o,v) \in Q_2$ and $e = u_1 \in U$ and $v \in \mathcal{U}$ and $\hat{q}_2 = (q_2, u_1) \in Q_0$ such that $(q,(d,u),(o,v),q_2) \in E$.

Define the set of accepting states as $\hat{F} = Q \times U$. Define now the automaton $\mathcal{A} = (\hat{Q}, \Sigma, \delta, \hat{F}, q_\perp)$. It is easy to see that $\mathcal{A}$ accepts the language L_R, hence L_R is indeed regular. Moreover, the construction above is computationally effective. By applying the usual power-set algorithm to $\mathcal{A}$, one gets a deterministic automaton recognizing L_R. ■

Furthermore, we will need the following maps.

Notation 1 (Erasing events in E_i): The projection $\theta : (U \cup D \cup O \cup E_i)^* \to (U \cup D \cup O)^*$ deletes elements of E_i; θ is a morphism such that $\theta(\epsilon) = \epsilon$, $\theta(a) = a$ if $a \in (U \cup D \cup O)$ and $\theta(e) = \epsilon$ for all $e \in E_i$.

Notation 2 (Erasing events not in E_i): The morphism $\theta_c : (U \cup D \cup O \cup E_i)^* \to E_i^*$ erases all occurrences of letters not in E_i, i.e. $\theta_c(\epsilon) = \epsilon$, $\theta_c(a) = \epsilon$ if $a \in (U \cup D \cup O)$, and $\theta_c(a) = a$ if $a \in E_i$.

Now we are ready to state the RW problem corresponding to Problem 1.

Problem 2 (RW counterpart of Problem 1): Assume that the specification language $K \subseteq E_i^* \cup E_i^\omega$ satisfies Assumption 1, and the sequential input-output map R is rational. Define the *Ramadge-Wonham problem with partial observations corresponding to Problem 1* as follows.

- *Controllable and uncontrollable events*
  Let the alphabet be $\Sigma = (U \cup D \cup O \cup E_i)$, let $\Sigma_c = U$ be the set of controllable events, and $\Sigma_{uc} = D \cup O \cup E_i$ be the set of uncontrollable events.
- *Observable and unobservable events*
  Let the set of observable events be $\Sigma_o = (U \cup D \cup O)$ and the set of unobservable events be $\Sigma_{uo} = E_i$.
- *Control requirements*
  Define the language of control requirements $K_s = \theta_c^{-1}(K_{safe})$, where K_{safe} is as in Assumption 1.
- *Plant language*
  Let the language of the plant G be the prefix closure of the language L_R from Definition 10, and let the marked language of G be L_R.

With the above alphabet Σ, plant G, requirements K_s and partitioning into controllable/uncontrollable and observable/unobservable events, find a non-blocking supervisor $S : \theta(\bar{L}_R) \to 2^{\Sigma_c}$ with partial observations such that the closed-loop system satisfies $L_m(G/S) \subseteq K_s$.

Proposition 2 (Regularity of K_s): If K_{safe} is regular, then the language K_s is regular, and its automaton can easily be computed from an automaton accepting K.

Problem 2 is a classical Ramadge-Wonham control problem with partial observations with well-known solution algorithms and tools [26]

B. From a supervisor to a controller

Below we formulate a procedure to extract a sequential controller from a supervisor solving Problem 2.

Definition 11 (Controller associated with a supervisor): Let $S : \theta(\bar{L}_R) \to 2^U$ be a non-blocking supervisor. A sequential controller $\phi : (O \times D)^* \to U$ *associated with* S is a sequential controller satisfying the following. For each collection of external inputs $d_i \in D$ and outputs $o_i \in O$, $i = 0,1,2,\ldots,k$, define

$$\begin{aligned} u_{i+1} &= \phi_S((d_1,o_1)(d_2,o_2)\cdots(d_i,o_i)) \\ w_i &= (u_1 d_1 o_1)(u_2 d_2 o_2)\cdots(u_i d_i o_i) \in \Sigma^* \end{aligned} \tag{3}$$

where $u_i \in U$ is the control input generated by ϕ. With the notation above, for all $i = 0,1,2,\ldots,k$.

$$[\forall j \le i : w_j \in \theta(L(G/S))] \implies u_{i+1} \in S(w_i) \tag{4}$$

In other words, a controller associated with a supervisor is a sequential controller which generates inputs which, if viewed as controllable events, are enabled by the supervisor. That is, if the controller generates inputs $u_1, u_2, \cdots u_{k+1}$ while reading $(d_1,o_1),(d_2,o_2)\cdots(d_i,o_i)$, then each u_i must be enabled by the supervisor S for the string $u_1 d_1 o_1 u_1 \cdots u_i d_i o_i$.

Proposition 3: For any non-blocking supervisor $S : \theta(\bar{L}_R) \to 2^U$ implementable by a finite-state automaton, there exists an associated sequential controller ϕ_S.

Proof: [Sketch of the proof of Proposition 3] Assume that the supervisor S can be represented by the automaton $A = (Q, \Sigma_o, \delta, F, q_0)$. That is, for any $v \in \theta(L(G/S))$, $\delta(q_o, v)$ is defined and $u \in S(v)$ if and only if $\delta(q_0, vu)$ is defined. Define the Moore-automaton $A_\phi = (Q_\phi, (D \times O), U, \delta_\phi, \lambda, q_\phi)$.

- $Q_\phi = (Q \times U) \cup \{\perp\}$ where $\perp \notin (Q \times U)$.
- The initial state is $q_\phi = (q_0, u)$, where $u \in U$ is chosen so that $\delta(q_0, u)$ is defined.

- The state-transition map $\delta_\phi : Q_\phi \times (D \times O) \to Q_\phi$ is defined as follows. For all $(d, o) \in (D \times O)$, $(q, u) \in Q \times U$, if $\delta(q, udo)$ is defined, then define $\delta_\phi((q, u), (d, o)) = (\delta(q, udo), \hat{u})$ where $\hat{u} \in U$ is chosen so that $\delta(q, udo\hat{u})$ is defined. If no such input $\hat{u} \in U$ exists, or $\delta(q, udo)$ is not defined, then define $\delta_\phi((q, u), (d, o)) = \perp$. Let $\delta_\phi(\perp, (d, o)) = \perp$.
- The readout map $\lambda : Q_\phi \to U$ is defined by $\lambda((q, u)) = u$ for all $(q, u) \in Q_\phi$, and $\lambda(\perp) = u \in U$ for some arbitrarily chosen $u \in U$.

Note that q_ϕ is well-defined, due to non-blockingness of S and the definition of L_R. Consider the input-output map $\phi : (D \times O)^* \to U$ realized by A_ϕ. It can be shown that ϕ satisfies Definition 11. ■

Notice that the proof of Proposition 3 does not yield a unique controller asscociated with the supervisor.

C. Correctness of the transformation

Now we are ready to state the main theorem, relating solutions of Problem 1 with those of Problem 2.

Theorem 1: If the supervisor S is a solution to Problem 2, then any sequential controller ϕ associated with S is a solution to Problem 1, and at least one such sequential controller exists.

Proof: [Sketch] The second part of the statement follows from Proposition 3. Let now ϕ be a sequential controller associated with S. We will show that $L(R/\phi) \subseteq K$, where $K = K_{safe} \cup \lim(K_{safe})$. To this end, it is enough to show

$$\forall v \in D^* : B(R/\phi)(v) \subseteq K_{safe} \qquad (5)$$

Since K_{safe} is prefix closed, S is non-blocking, and $L_m(G/S) \subseteq \theta_c^{-1}(K_{safe})$, it follows that $L(G/S) \subseteq \theta_c^{-1}(K_{safe})$ Hence, for (5), it is enough to show that

$$\forall v \in D^* : B(R/\phi)(v) \subseteq \theta_c(L(G/S))$$

The latter relation follows from the construction of L_R and the assumption that ϕ is a sequential controller asscociated with the supervisor S. ■

V. Conclusions

We have presented a novel framework for control of discrete-event systems, which allows for explicit modeling of inputs and output and active generation of events by the controller. We also show that under suitable assumptions the proposed control problem can be solved using the classical RW framework. In [23] the results of the paper are presented in more detail, including an example of practical importance.

Future research direction include tackling a more general version of the proposed control problem by using games on automata [14]. In addition, we would like to investigate the possibility of reducing computational complexity of the solution. Furthermore, we plan more case studies for validating the theory on industrial examples

Acknowledgments The authors thank R. Schiffelers and P. Collins for useful discussion and remarks.

References

[1] Silvano Balemi. Input/output discrete event processes and communication delays. *Discrete Event Dynamic Systems*, 4(1):41–85, 1994.

[2] J. Berstel. *Transductions and Context-Free Languages*. Teubner, Stuttgart, 1979.

[3] F. Charbonnier, H. Alla, and R. David. Discrete-event dynamic systems. *IEEE Trans. Control Sys. Technology*, 7(2):175–187, 1999.

[4] J.E.R. Cury, B.H. Krogh, and T. Niinomi. Synthesis of supervisory controllers for hybrid systems based on approximating automata. *IEEE Trans. Automatic Control*, 43(4), 1998.

[5] M.H. de Queiroz and J.E.R. Cury. Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *6th Int. Workshop on Discrete Event Systems*, 2002.

[6] M.D. Di Benedetto and A. Sangiovanni-Vincentelli. Model matching for finite-state machines. *IEEE Trans. Automatic Control*, 46(11), 2001.

[7] P. Dietrich, R. Malik, W. M. Wonham, and B. A. Brandin. Implementation considerations in supervisory control. In *Synthesis and Control of Discrete Event Systems*, pages 185–201. Kluwer Academic Publishers, 2002.

[8] Liang Du, S.L. Ricker, and P. Gohari. Decentralized supervisory control and communication for reactive discrete-event systems. In *American Control Conference, 2006*, page 6 pp., 2006.

[9] Samuel Eilenberg. *Automata, Languages and Machines*. Academic Press, New York, London, 1974.

[10] M. Fabian and A. Hellgren. PLC-based implementation of supervisory control for discrete event systems. In *Proc. 37th IEEE Conference on Decision and Control*, volume 3, 1998.

[11] F. Gécseg and I Peák. *Algebraic theory of automata*. Akadémiai Kiadó, Budapest, 1972.

[12] C. H. Golaszewski and P. J. Ramadge. Control of discrete event processes with forced events. In *Decision and Control, 1987. 26th IEEE Conference on*, volume 26, pages 247–251, 1987.

[13] J.M.E Gonzalez, A.E.C da Cunha, J.E.R. Cury, and B.H Krogh. Supervision of event-driven hybrid systems: Modeling and synthesis. In *Hybrid Systems: Computation and Control*, volume LNCS 2034, pages 247 – 260, 2001.

[14] E. Grädel, W. Thomas, and T. Wilke. *Automata, Logic and Infinite Games*, volume LNCS 2500. Springer, 2002.

[15] J. Huang and R. Kumar. Optimal nonblocking directed control of discrete event systems. In *Proc. American Control Conference*, 2007.

[16] X.D. Koutsoukos, P.J. Antsaklis, J.A. Stiver, and M.D. Lemmon. Supervisory control of hybrid systems. *Proceedings of the IEEE*, 88(7):1026–1049, 2000.

[17] Jan Lunze. Relations between networks of standard automata and networks of i/o automata. In *9th International Workshop on Discrete Event Systems*, pages 425 – 430, 2008.

[18] N. Lynch and M. Tuttle. An introduction to input/output automata. *CWI-Quarterly*, 2(3):219–246, 1989.

[19] P. Mahdavinezhad, P. Gohari, and A.G. Aghdam. Supervisory control of discrete-event systems with output: Application to hybrid systems. In *Proc. American Control Conference*, pages 4291–4296, 2007.

[20] P. Malik. *From Supervisory Control to Nonblocking Controllers for Discrete Event Systems*. PhD thesis, Dept. of Computer Science, University of Kaiserslautern, 2003.

[21] A. Morgenstern and K. Schneider. Synthesizing deterministic controllers in supervisory control. In *Informatics in Control, Automation and Robotics II*, 2007.

[22] S. Perk, T. Moor, and K. Schmidt. Controller synthesis for an i/o-based hierarchical system architecture. pages 474–479, May 2008.

[23] M. Petreczky, R. Theunissen, R. Su, D.A. van Beek, and J.E. van Schuppen J.H.and Rooda. Control of input-output discrete-event systems. Technical Report 2008-12, Eindhoven University of Technology, Systems Engineering, 2008.

[24] M. Petreczky, D. A. van Beek, and J. E. Rooda. Supervisor for toner error-handling. Technical report, Eindhoven University of Technology, Systems Engineering, 2008.

[25] R. Theunissen, R. Schiffelers, D.A. van Beek, and J.E. Rooda. Supervisory control synthesis for a patient support system. Technical Report 2008-08, Eindhoven University of Technology, 2008.

[26] W.M. Wonham. Supervisory control of discrete-event systems. Lecture notes, `http://www.control.utoronto.ca/~wonham`.